

Lab Exercise 5– Terraform Variables with Command Line Arguments

Objective:

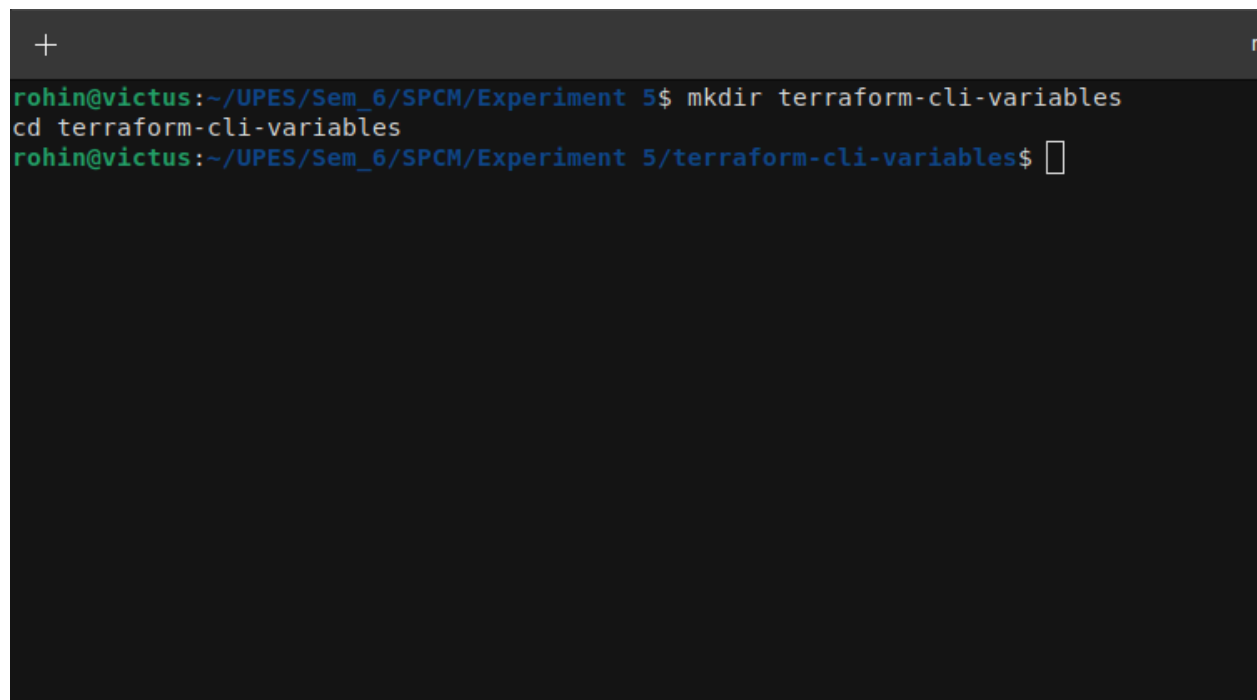
Learn how to pass values to Terraform variables using command line arguments.

Prerequisites:

- Terraform installed on your machine.
- Basic knowledge of Terraform variables.

Steps:

1. Create a Terraform Directory:

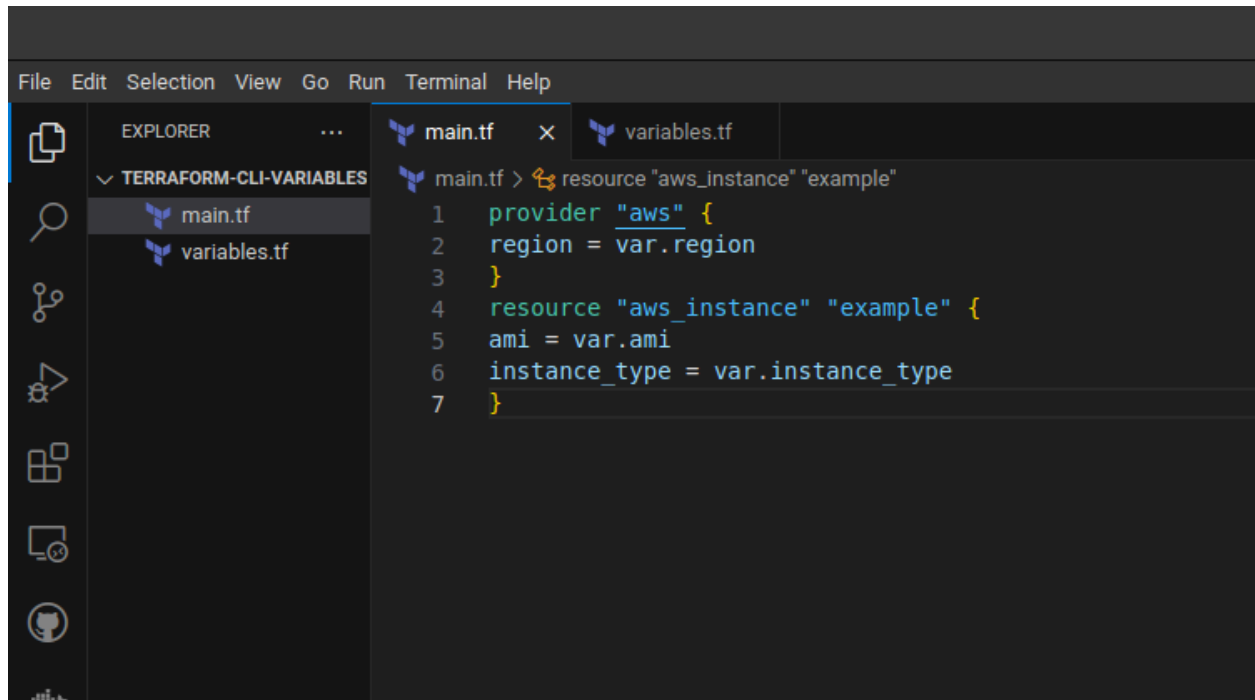
A terminal window with a dark background and light-colored text. The prompt is 'rohin@victus:~/UPES/Sem_6/SPCM/Experiment 5\$'. The user enters 'mkdir terraform-cli-variables' and presses enter. The prompt changes to 'rohin@victus:~/UPES/Sem_6/SPCM/Experiment 5/terraform-cli-variables\$'. The user then enters 'cd terraform-cli-variables' and presses enter. The prompt changes to 'rohin@victus:~/UPES/Sem_6/SPCM/Experiment 5/terraform-cli-variables\$' with a cursor at the end.

```
rohin@victus:~/UPES/Sem_6/SPCM/Experiment 5$ mkdir terraform-cli-variables
rohin@victus:~/UPES/Sem_6/SPCM/Experiment 5$ cd terraform-cli-variables
rohin@victus:~/UPES/Sem_6/SPCM/Experiment 5/terraform-cli-variables$
```

2. Create Terraform Configuration Files:

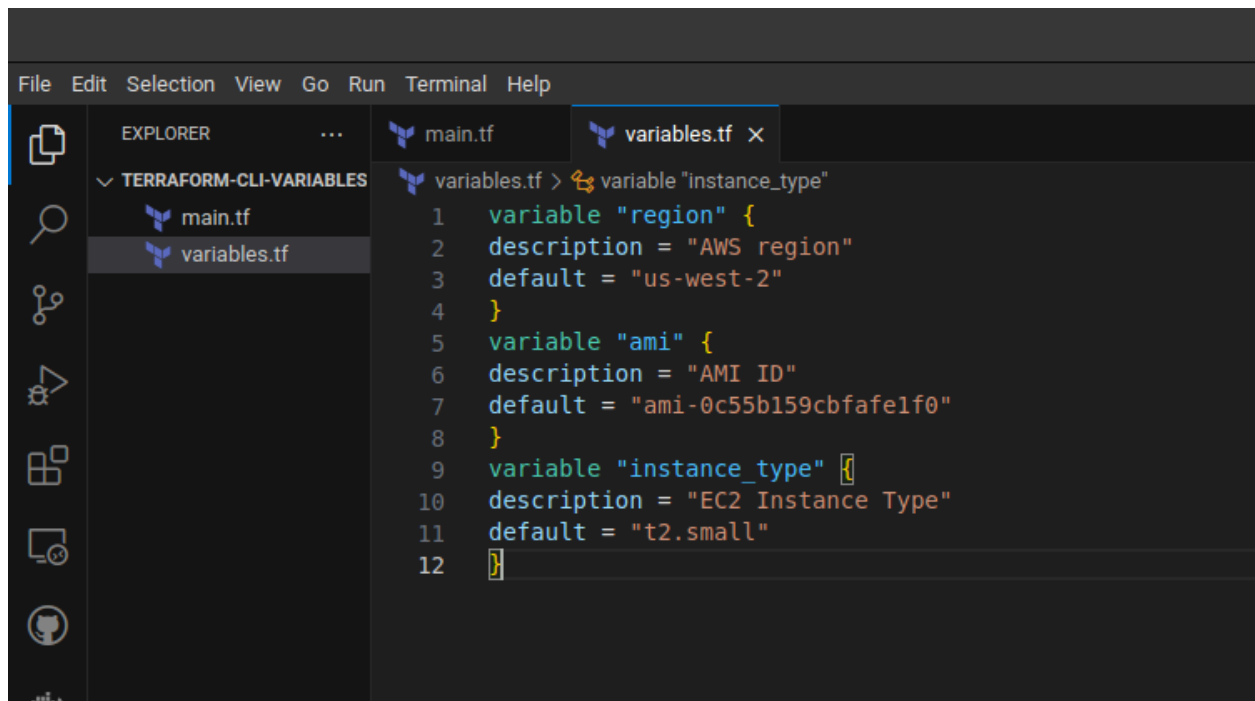
- Create a file named main.tf:

main.tf



•Create a file named variables.tf:

variables.tf



3. Use Command Line Arguments:

- Open a terminal and navigate to your Terraform project directory.
- Run the terraform init command:

```
+ rohin@victus: ~
rohin@victus:~/UPES/Sem_6/SPCM/Experiment 5/terraform-cli-variables$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.35.0...
- Installed hashicorp/aws v5.35.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
rohin@victus:~/UPES/Sem_6/SPCM/Experiment 5/terraform-cli-variables$
```

- Run the terraform apply command with command line arguments to set variable values:

```
+ rohin@victus: ~/UPES/Sem_6/SPCM/Experiment 5/terraform-cli-variables
rohin@victus:~/UPES/Sem_6/SPCM/Experiment 5/terraform-cli-variables$ terraform apply -var 'region=us-east-1' -var 'ami=ami-0c55b159cbfafef0' -var 'instance_type=t2.micro'

Terraform used the selected providers to generate the following execution plan. Resource actions are
indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.example will be created
+ resource "aws_instance" "example" {
  + ami                    = "ami-0c55b159cbfafef0"
  + arm                   = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone       = (known after apply)
  + cpu_core_count          = (known after apply)
  + cpu_threads_per_core    = (known after apply)
  + disable_api_stop        = (known after apply)
  + disable_api_termination = (known after apply)
  + ebs_optimized           = (known after apply)
  + get_password_data       = false
  + host_id                 = (known after apply)
  + host_resource_group_arn = (known after apply)
  + iam_instance_profile    = (known after apply)
  + id                      = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
  + instance_lifecycle      = (known after apply)
  + instance_state          = (known after apply)
}
```

4. Test and Verify:

- Observe how the command line arguments dynamically set the variable values during the apply process.
- Access the AWS Management Console or use the AWS CLI to verify the creation of resources in the specified region.

5. Clean Up:

After testing, you can clean up resources:

```
`terraform destroy`  
`
```

Confirm the destruction by typing yes.

6. Conclusion:

This lab exercise demonstrates how to use command line arguments to set variable values dynamically during the terraform apply process. It allows you to customize your Terraform deployments without modifying the configuration files directly. Experiment with different variable values and observe how command line arguments impact the infrastructure provisioning process.